**Security assessment and testing programs perform regular checks to ensure that adequate security controls are in place and that they effectively perform their assigned functions**

three major components of a security assessment program:

■ Security tests

■ Security assessments

■ Security audits

**Security Testing**

*Security tests* verify that a control is functioning properly. These tests include automated scans, tool-assisted penetration tests and manual attempts to undermine security. Security testing should take place on a regular schedule, with attention paid to each of the key security controls protecting an organization.

Many security testing programs begin on a haphazard basis, with security professionals simply pointing their fancy new tools at whatever systems they come across first. Experimentation with new tools is fine, but security testing programs should be carefully designed and include rigorous, routine testing of systems using a risk-prioritized approach.

**Security Assessments**

*Security assessments* are comprehensive reviews of the security of a system, application, or other tested environment. During a security assessment, a trained information security professional performs a risk assessment that identifies vulnerabilities in the tested environment that may allow a compromise and makes recommendations for remediation, as needed. Security assessments normally include the use of security testing tools but go beyond automated scanning and manual penetration tests. They also include a thoughtful review of the threat environment, current and future risks, and the value of the targeted environment. The main work product of a security assessment is normally an assessment report addressed to management that contains the results of the assessment in nontechnical language and concludes with specific recommendations for improving the security of the tested environment.

**Security Audits**

*Security audits* use many of the same techniques followed during security assessments but must be performed by independent auditors.

*Internal audits* are performed by an organization’s internal audit staff and are typically intended for internal audiences. The internal audit staff performing these audits normally have a reporting line that is completely independent of the functions they evaluate. In many organizations, the Chief Audit Executive reports directly to the President, Chief Executive Officer, or similar role. The Chief Audit Executive may also have reporting responsibility directly to the organization’s governing board.

*External audits* are performed by an outside auditing fi rm. These audits have a high degree of external validity because the auditors performing the assessment theoretically have no conflict of interest with the organization itself. There are thousands of firms who perform external audits, but most people place the highest credibility with the so-called “Big Four” audit firms:

■ Ernst & Young ■ Deloitte & Touche ■ PricewaterhouseCoopers ■ KPMG

Auditors generally have ***carte blanche***access to all information within an organization and security staff should comply with those requests, consulting with management as needed.

**Just to be clear on terminology, vulnerability assessments as they are described in this chapter are actually security *testing* tools, not security *assessment* tools. They probably should be called vulnerability tests for linguistic consistency**

**There are three main categories of vulnerability scans: network discovery scans, network vulnerability scans, and web application vulnerability scan**

**Network Discovery Scanning**

Network discovery scanning uses a variety of techniques to scan a range of IP addresses, searching for systems with open network ports. Network discovery scanners do not actually probe systems for vulnerabilities but provide a report showing the systems detected on a network and the list of ports that are exposed through the network and server firewalls that lie on the network path between the scanner and the scanned system. Network discovery scanners use many different techniques to identify open ports on remote systems. Some of the more common techniques are as follows:

**TCP SYN Scanning** Sends a single packet to each scanned port with the SYN flag set. This indicates a request to open a new connection. If the scanner receives a response that has the SYN and ACK flags set, this indicates that the system is moving to the second phase in the three-way TCP handshake and that the port is open. **TCP SYN scanning is also known as “half-open” scanning**

**TCP Connect Scanning** Opens a full connection to the remote system on the specified port. This scan type is used when the user running the scan does not have the necessary permissions to run a half-open scan

**TCP ACK Scanning** Sends a packet with the ACK flag set, indicating **that it is part of an open connection**

**Xmas Scanning** Sends a packet with the FIN, PSH, and URG flags set. A packet with so many flags set is said to be “lit up like a Christmas tree,” leading to the scan’s name.

**The most common tool used for network discovery scanning is an open source tool called nmap**

For ports where nmap detects a result, it provides the current status of that port:

**Open** The port is open on the remote system and there is an application that is actively accepting connections on that port.

**Closed** The port is accessible on the remote system, meaning that the firewall is allowing access, but there is no application accepting connections on that port.

**Filtered** Nmap is unable to determine whether a port is open or closed because a firewalls interfering with the connection attempt

**Network Vulnerability Scanning**

*Network vulnerability scans* go deeper than discovery scans. They don’t stop with detecting open ports but continue on to probe a targeted system or network for the presence of known vulnerabilities. These tools contain databases of thousands of known vulnerabilities, along with tests they can perform to identify whether a system is susceptible to each vulnerability in the system’s database.

In some cases, the scanner may not have enough information to conclusively determine that a vulnerability exists and it reports a vulnerability when there really is no problem. This situation is known as a ***false positive***report and is sometimes seen as a nuisance to system administrators. Far more dangerous is when the vulnerability scanner misses a vulnerability and fails to alert the administrator to the presence of a dangerous situation. This error is known as **a *false negative***report

By default, network vulnerability scanners run **unauthenticated scans**. They test the target systems without having passwords or other special information that would grant the scanner special privileges. This allows the scan to run from the perspective of an attacker but also limits the ability of the scanner to fully evaluate possible vulnerabilities. One way to improve the accuracy of the scanning and reduce false positive and false negative reports is to perform ***authenticated scans* of systems**

Important port numbers

FTP 21

SSH 22

Telnet 23

SMTP 25

DNS 53

HTTP 80

POP3 110

NTP 123

HTTPS 443

Microsoft SQL Server 1433

Oracle 1521

H.323 1720

PPTP 1723

RDP 3389

**Web Vulnerability Scanning**

Nessus is an example of a hybrid tool that can perform both types of scan (network &Web application)

**Penetration Testing**

The process may include the following:

■ Performing basic reconnaissance to determine system function (such as visiting websites hosted on the system)

■ Network discovery scans to identify open ports

■ Network vulnerability scans to identify unpatched vulnerabilities

■ Web application vulnerability scans to identify web application flaws

■ Use of exploit tools to automatically attempt to defeat the system security

■ Manual probing and attack attempts.

**Penetration testers commonly use a tool called *Metasploit* to automatically execute.** uses a scripting language.

**White Box Penetration Test** Provides the attackers **with detailed information about the systems they target.** This **bypasses many of the reconnaissance steps** that normally precede attacks, shortening the time of the attack and increasing the likelihood that it will find security flaws.

**Black Box Penetration Test** Does not provide attackers with any information prior to the attack. This simulates an external attacker trying to gain access to information about the business and technical environment before engaging in an attack.

**Gray Box Penetration Test** Also known as partial knowledge tests, these are sometimes chosen to balance the advantages and disadvantages of white and black box penetration tests. This is particularly common when black box results are desired, but costs or time constraints mean that some knowledge is needed to complete the testing

**Code Review and Testing**

One of the most critical components of a software testing program is conducting code review and testing. These procedures provide third-party reviews of the work performed by developers before moving code into a production environment. Code reviews and tests may discover security, performance, or reliability flaws in applications before they go live and negatively impact business operations

***Code review* is the foundation of software assessment programs**. During a code review, also known as a “peer review,” developers other than the one who wrote the code review it for defects

**The most formal code review processes, known as Fagan inspections, follow a rigorous review and testing process with six steps**

**1.** Planning **2.** Overview **3.** Preparation **4.** Inspection **5.** Rework **6.** Follow-up

Each of these steps has well-defined entry and exit criteria that must be met before the process may formally

transition from one stage to the next.

The Fagan inspection level of formality is normally found only in highly restrictive environments where code

flaws may have catastrophic impact. Most organizations use less rigorous processes using code peer review measures that include the following:

■ Developers walking through their code in a meeting with one or more other team members

■ A senior developer performing manual code review and signing off on all code before moving to production

■ Use of automated review tools to detect common application flaws before moving to production

Each organization should adopt a code review process that suits its business requirements and software development culture.

**Dynamic testing may include the use of *synthetic transactions* to verify system performance**

**Fuzz Testing**

*Fuzz testing* is a specialized dynamic testing technique that provides many different types of input to software to stress its limits and find previously undetected flaws. Fuzz testing software supplies invalid input to the software, either randomly generated or specially crafted to trigger known software vulnerabilities. The fuzz tester then monitors the performance of the application, watching for software crashes, buffer overflows, or other undesirable and/or unpredictable outcomes. There are two main categories of fuzz testing.

**Mutation (Dumb) Fuzzing** Takes previous input values from actual operation of the software and manipulates (or mutates) it to create fuzzed input. It might alter the characters of the content, append strings to the end of the content, or perform other data manipulation techniques.

**Generational (Intelligent) Fuzzing** Develops data models and creates new fuzzed input based on an understanding of the types of data used by the program.

**The process of slightly manipulating the input is known as *bit flipping*.**

**Interface Testing**

*Interface testing* is an important part of the development of complex software systems. In many cases, multiple teams of developers work on different parts of a complex application that must function together to meet business objectives. The handoffs between these separately developed modules use well-defined interfaces so that the teams may work independently. Interface testing assesses the performance of modules against the interface specifications to ensure that they will work together properly when all the development efforts are complete

**Application Programming Interfaces (APIs)** Offer a standardized way for code modules to interact and may be exposed to the outside world through web services. Developers must test APIs to ensure that they enforce all security requirements.

**User Interfaces (UIs)** Examples include graphic user interfaces (GUIs) and command-line interfaces. UIs provide end users with the ability to interact with the software. Interface tests should include reviews of all user interfaces to verify that they function properly.

**Physical Interfaces** Exist in some applications that manipulate machinery, logic controllers, or other objects in the physical world. Software testers should pay careful attention to physical interfaces because of the potential consequences if they fail.

**Misuse Case Testing**

In some applications, there are clear examples of ways that software users might attempt to misuse the application. For example, users of banking software might try to manipulate input strings to gain access to another user’s account. They might also try to withdraw funds from an account that is already overdrawn. Software testers use a process known as *misuse case testing* or *abuse case testing* to evaluate the vulnerability of their software to these known risks.

In misuse case testing, testers first enumerate the known misuse cases. They then attempt to exploit those use cases with manual and/or automated attack techniques.

Software testing professionals often conduct a *test coverage analysis* to estimate the degree of testing conducted against the new software. The test coverage is computed using the following formula:

***test coverage=number of use cases tested/total number of use cases…***